NWU®

Requirements for this paper/Benodigdhede vir hierdie vraestel:			
Answer scripts/ Antwoordskrifte:	X	Multi-choice cards (A5)/ Multikeusekaarte (A5):	
Attendance slips (Fill-in paper)/ Presensiestrokies (Invulvraestel):		Multi-choice cards (A4)/ Multikeusekaarte (A4):	
Scrap paper/ Rofwerkpapier:		Graph paper/ Grafiekpapier:	

Calculators/Sakrekenaars:	Yes/Ja
Other resources/Ander hulpmiddels:	
A/ A	
B/ B	
C/ C	

Type of Assessment/ Tipe Assessering:	Test 2	Qualification/ Kwalifikasie:	BSc
Module code/ Modulekode:	ITRI616	Duration/ Tydsduur:	1 hour / 1 uur
Module description/ Module beskrywing:	Artificial Intelligence	Max/ Maks:	53
Examiner(s)/ Eksaminator(e):	MR. J.J. Greeff (VC)	Date/ Datum:	03/06/2021
Internal/Interne Moderator(s):	PROF. J. V. (TINY) DU TOIT (PC)	Time/ Tyd:	1:1

Submission of answer scripts/Inhandiging van antwoordskrifte:

Question 1 (Search Algorithms)

1.1. What types of queues are used in search algorithms, and for what types of search would you use each type of queue? **[6]**

Watter soort toue word in soekalgoritmes gebruik en vir watse algoritmes sal jy elke tipe gebruik?.

One mark for each type of queue and one mark for each algorithm it is used for

A priority queue first pops the node with the minimum cost according to some evaluation function, f. It is used in best-first search.

A FIFO queue or first-in-first-out queue first pops the node that was added to the queue first; we shall see it is used in breadth-first search

A FIFO queue or first-in-first-out queue first pops the node that was added to the queue first; we shall see it is used in breadth-first search

1.2. Explain the four ways which can be used to evaluate the performance of a search algorithm. **[8]**

Verduidelik die vier maniere waarop die werkverrigting van 'n soekalgoritme gemeet kan word.

One mark for each measure and one mark for each short description

Completeness: Is the algorithm guaranteed to find a solution when there is one, and to correctly report failure when there is not?

Cost optimality: Does it find a solution with the lowest path cost of all solutions?

Time complexity: How long does it take to find a solution? This can be measured in seconds, or more abstractly by the number of states and actions considered.

Space complexity: How much memory is needed to perform the search?

1.3 How would one go about measuring the time and space complexity of:

a) A state space where the graph is represented with an explicit data structure. **[2]**

b) A state space where the graph is represented implicitly. **[3]**

Hoe sal mens die tyd en spasie kompleksiteit bereken van:

a) n soekruimte waar die grafiek eksplisiet verteenwoordig word.
b) n soekruimte waar die grafiek implisiet verteenwoordig word.

a. In theoretical computer science, the typical measure is the size of the state-space graph, |V|+|E|, where |V| is the number of vertices (state nodes) of the graph and |E| is the number of edges (distinct state/action pairs). This is appropriate when the graph is an explicit data structure, such as the map of Romania.

One mark for each |V| and |E| and their description

b. For an implicit state space, complexity can be measured in terms of d, the depth or number of actions in an optimal solution; m, the maximum number of actions in any path; and b, the branching factor or number of successors of a node that need to be considered.

One mark for each d, m and b and their description

1.4 What is the difference between Graph search and Tree-like search and when would you use each? **[2]**

Wat is die verskil tussen grafiek soek en boom soek en wanneer sal jy elkeen gebruik?

We call a search algorithm a graph search if it checks for redundant paths and a tree-like search if it does not check (1). Graph search should be used when there are multiple paths that can reach each node and tree search can be used when the problem is of such a nature that nodes can only be reached in a set order (like a manufacturing problem) (1).

1.5 Why do we use standardized problems for search algorithms? **[2]**

Hoekom gebruik ons gestandardiseerde probleme met soek algoritmes?

To illustrate the function of different algorithms (1) as well as to use standardised tests to measure the performance of algorithms (1).

1.6 Name and describe four examples of real world problems that can be approached using search algorithms. **[8]**

Noem en beskryf vier voorbeelde van egte wereld probleme wat benader kan word met die hulp van soek algoritmes.

One mark for each problem and one mark for each description.

Touring problems describe a set of locations that must be visited, rather than a single goal destination. The traveling salesperson problem (TSP) is a touring problem in which every city on a map must be visited. The aim is to find a tour with cost <C (or in the optimization version, to find a tour with the lowest cost possible).

A **VLSI layout** problem requires positioning millions of components and connections on a chip to minimize area, minimize circuit delays, minimize stray capacitances, and maximize manufacturing yield.

Robot navigation is a generalization of the route-finding problem described earlier. Rather than following distinct paths (such as the roads in Romania), a robot can roam around, in effect making its own paths.

Automatic assembly sequencing of complex objects (such as electric motors) by a robot has been standard industry practice since the 1970s. Algorithms first find a feasible assembly sequence and then work to optimize the process. Minimizing the amount of manual human labor on the assembly line can produce significant savings in time and cost.

Question 2 (Uninformed Search Algorithms)

2.1. Given a tree with d = 6 and b = 8, calculate and compare the number of nodes that will be searched using Iterative Deepening Search and Breadth First Search. When would you use one search algorithm over the other? **[6]**

Gegewe n boom met d = 6 en b = 8, bereken en vergelyk die hoeveelheid nodes wat besoek sal word deur n Iterative Verdieping Soek Algoritme en n Breedte Eerste Soek Algoritme. Wanneer sal jy die een algoritme bo die ander een verkies?

N(IDS) = $(d)\ b^1 + (d-1)b^2 + (d-2)\ b^3 \ldots.\ b^d$ One mark

$= 6x8 + 5x8^2 + 4x8^3 + 3x8^4 + 2x8^5 + 1x8^6$

= 48 + 320 + 2048 + 12288 + 65536 + 262144

= 342384 One mark

N(BFS) = $b^1 + b^2 + b^3 \ldots b^d$ One mark

$= 8 + 8^2 + 8^3 + 8^4 + 8^5 + 8^6$

= 8 + 64 +512 + 4096 + 32768 + 262144

= 299592 One mark

IDS is generally preferred because it has a much lower memory requirement due to it not keeping all of the nodes in memory the way BFS does. That said, it can be a little bit slower due to the repetition in searching the nodes in the lower depths. In problems where the depth of the goal node is known it can be beneficial to use BFS, but generally IDS is used when the depth of the goal is not known.

Two marks for discussion of when to use each algorithm.

Question 3 (Informed Search Algorithms)

3.1. What is the difference between informed and uninformed search algorithms? **[1]**

Wat is die verskil tussen ingeligte en oningeligte soek algoritmes?

Informed search algorithms attempt to speed up the search process by using domain specific hints about where the goal(s) may be in the form of the heuristic evaluation function.

3.2 What is meant by the admissibility of a heuristic? **[1]**

Wat word bedoel met toelaatbaarheid van n heuristiek?

an admissible heuristic is one that never overestimates the cost to reach a goal. (1)

3.3 Using the map of Romania and the straight line distances in the provided table, perform an A* search from Oradea to Urziceni. Assume that your algorithm is checking for redundant paths and won't add nodes into the tree if a shorter path already exists to that node. **[14]**

Gebruik die kaart van Romania en die gegewe reguit lyn afstande in die tabel om n A Soek te doen van Oradea tot Urziceni. Jy kan aanneem dat jou algoritme uitkyk vir oorbodige roetes en sal nie nodes by las in die boom as n korter roete klaar in die boom bestaan nie.*

One mark for each correct calculation, two marks for correct route found

Oradea = 0 + 434 = 434 km

Zerind = 71 + 433 = 504 km

Sibiu = 151 + 298 = 449 km

Arad = 291 + 431 = 722 km

Rimnicu Vilcea = 231 + 255 = 486 km

Fagaras = 250 + 198 = 448 km

Bucharest = 461 + 66 = 527 km

*note, although bucharest is close to the goal, Rimnicu Vilcea has the lowest f cost, so it will get expanded next

Craiova = 377 + 235 = 612 km

Pitesti = 328 + 151 = 479 km

*note, pitesti would be expanded next, but has no child nodes that don't already have shorter paths to them. The next lowest f value is this zerind which adds a second node for Arad because it is shorter than the existing node

Arad = 146 + 431 = 577 km

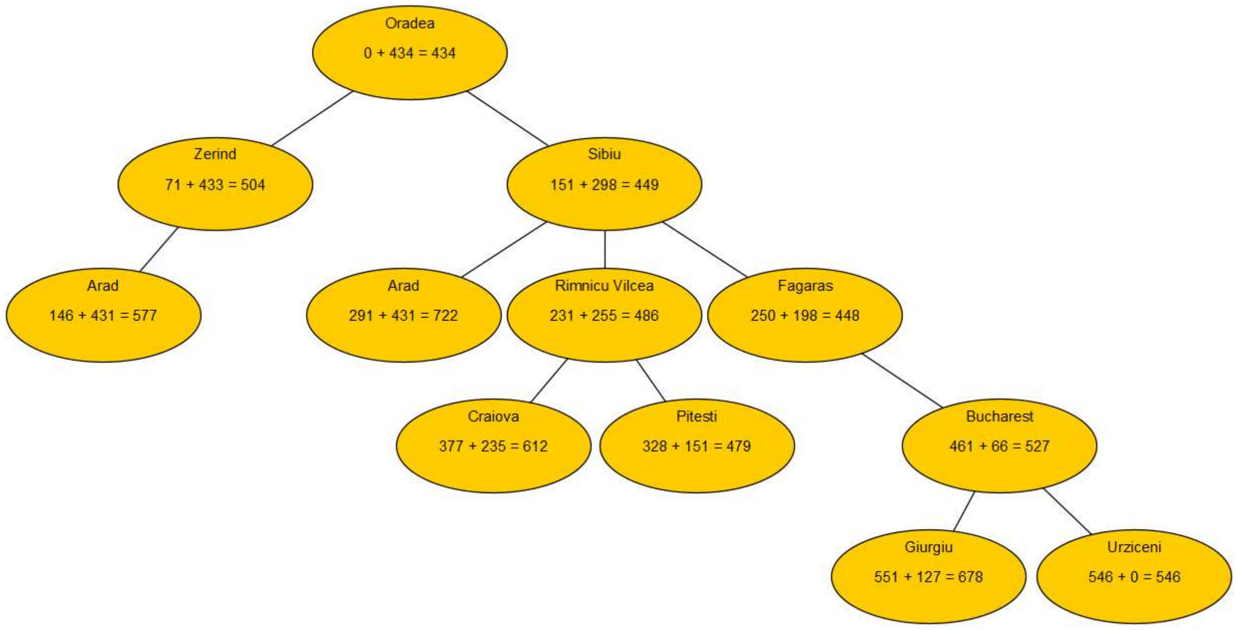
*note, now Bucharest is the lowest cost node and can be expanded

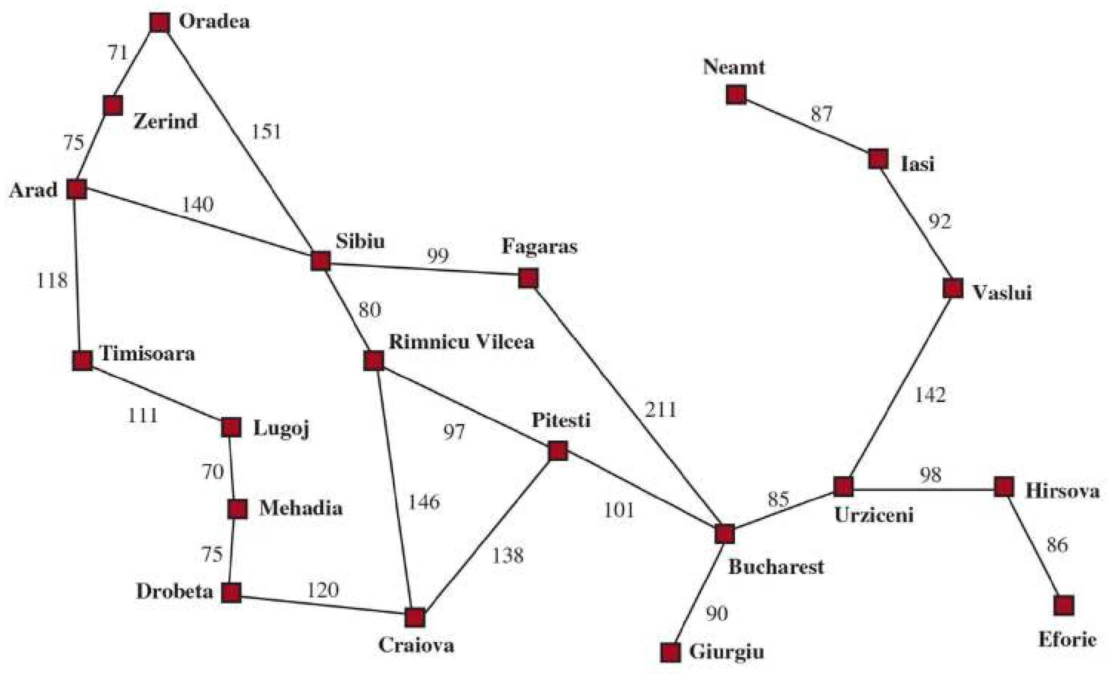
Giurgiu = 551 + 127 = 678 km

Urziceni = 546 + 0 = 546 km

The route calculated is therefore:

Oradea, Sibiu, Fagaras, Bucharest, Urziceni





SLD

Arad	431	Mehadia	317
Bucharest	66	Neamt	214
Craiova	235	Oradea	434
Drobeta	325	Pitesti	151
Efore	131	Rimnicu Vilcea	255
Fagaras	198	Sibiu	298
Giurgiu	127	Timisoara	404
Hirsova	84	Urziceni	0
Iasi	172	Vaslui	118
Lugoj	322	Zerind	433

TOTAL/TOTAAL: 53

File reference: 8.1.7.2.2